Metodología para el diseño multilenguaje de sistemas de control empotrados sobre FPGAs

E. del Toro
Centro de Investigaciones en Microelectrónica CUJAE
Ciudad de La Habana, Cuba
ernesto@electrica.cujae.edu.cu

S. Sánchez-Solano
Instituto de Microelectrónica de Sevilla. CSIC
41092 Sevilla, España
santiago@imse-cnm.csic.es

A. Cabrera
Dpto. de Automática y Computación. CUJAE
Ciudad de La Habana, Cuba
alex@electrica.cujae.edu.cu

A. Barriga
Instituto de Microelectrónica de Sevilla. CSIC
41092 Sevilla, España
barriga@imse-cnm.csic.es

Resumen

Este trabajo describe una estrategia de codiseño hardware-software para la implementación de sistemas de control difusos en FPGAs. Su principal aportación radica en que permite el desarrollo conjunto de los componentes hardware y software y posibilita la verificación y diseño eficiente de este tipo de sistema. El flujo de diseño propuesto combina las herramientas específicas para la realización de sistemas de inferencia difusos incluidas en *Xfuzzy* con herramientas de síntesis e implementación de FPGAs de Xilinx y herramientas de modelado y simulación del entorno Matlab.

1. Introducción

El diseño de los sistemas electrónicos actuales recurre en gran medida al uso de arquitecturas programables y requiere, en muchas ocasiones, que el diseñador combine elementos hardware y software para implementar el sistema de forma eficiente. Para facilitar el desarrollo conjunto de los elementos hardware y software de un sistema se utilizan técnicas de codiseño hardware-software.

En la realización de sistemas complejos es muy frecuente la utilización de varios lenguajes para describir distintos tipos de funcionalidades. La existencia de este tipo de sistema heterogéneo se debe a que numerosos estudios han demostrado que hasta la fecha no existe un lenguaje de descripción único, capaz de soportar las diferentes etapas del ciclo de desarrollo de un sistema (especificación, síntesis, verificación e implementación) [1],[2],[4],[8],[11].

El avance en la fabricación de circuitos integrados ha permitido la integración de cada vez más funcionalidad en un solo chip. Al utilizar los recursos disponibles en los FPGA actuales, es posible implementar de forma rápida sistemas de tipo SoPC (*System on Programmable Chip*). Pero, para sacar partido a estas tecnologías, se hace imprescindible la utilización de nuevos métodos de diseño que permitan elaborar un producto fiable en un tiempo razonable.

En este trabajo se describe el proceso de diseño y verificación de un sistema heterogéneo de control difuso aplicado al aparcamiento de un vehículo autónomo. El flujo de diseño empleado combina herramientas específicas para la realización de sistemas de inferencia difusos incluidas en *Xfuzzy*, herramientas de síntesis e implementación de FPGAs de Xilinx y herramientas de modelado y simulación del entorno Matlab.

El trabajo está estructurado como sigue. En la sección 2 se introducen las herramientas utilizadas en la metodología propuesta. La sección 3 describe una novedosa técnica de diseño para la realización hardware de sistemas de control difusos implementados en FPGAs. La sección 4 explica los pasos requeridos para la realización conjunta de un sistema hardware-software, así como la realización de funciones complementarias que facilitan la realización y el diseño de este sistema utilizando Matlab. Para finalizar, la sección 5 brinda las conclusiones de este trabajo.

2. Diseño de sistemas multilenguaje

El problema de la realización de un sistema multilenguaje puede ser abordado de dos formas:

Este trabajo ha sido parcialmente financiado por los proyectos FP7-IST-248858 de la Comisión Europea, TEC2008-04920 del Ministerio de Ciencia e Innovación y P08-TIC-03674 de la Junta de Andalucía.

la solución de tipo composicional y la variante basada en cosimulación.

La solución composicional trata de integrar una especificación parcial de cada una de las diferentes funcionalidades de un sistema en una representación unificada, que luego será utilizada para la verificación del comportamiento y el diseño global del sistema [6]. Los sistemas que en una primera fase se describen utilizando un lenguaje que trate de abarcar la funcionalidad completa, como Matlab, o variantes un poco más elaboradas, como aquellos realizados utilizando SystemC, pudieran constituir un ejemplo de soluciones de tipo composicional.

La variante basada en cosimulación consiste en la interconexión de los entornos de diseños asociados a las distintas funcionalidades de un sistema. Dado que no existe un lenguaje único que pueda ser utilizado para describir todas las funcionalidades que un sistema complejo necesita, los elementos que componen este sistema pueden estar descritos utilizando distintos lenguajes, como por ejemplo VHDL para describir una funcionalidad de tipo concurrente y C++ para describir mejor una funcionalidad secuencial.

En este trabajo se describe una estrategia de diseño multilenguaje que usa una variante basada en cosimulación. La metodología propuesta utiliza herramientas de diseño ampliamente conocidas e incluye la realización de algunos cambios que permiten la correcta integración y cosimulación del sistema completo. Siguiendo este método, los distintos componentes de sistema se han realizado utilizando distintos lenguajes y herramientas de desarrollo. En las diferentes etapas de diseño se ha utilizado cosimulación para validar la solución propuesta. La realización del trabajo ha requerido, asimismo, el desarrollo de mecanismos de comunicación inter-lenguaje y la puesta a punto de cada uno de los sub-sistemas. Las herramientas utilizadas en el flujo de diseño propuesto se describen a continuación.

2.1. Entorno de desarrollo *Xfuzzy*

Xfuzzy es un entorno de diseño que facilita el desarrollo de sistemas de inferencia difusos. Está compuesto por una serie de herramientas que permiten definir, verificar y optimizar sistemas difusos. El entorno de desarrollo incluye además herramientas de síntesis que facilitan su implementación como componentes software o hardware [14]. Se pueden describir controladores difusos complejos utilizando una especificación jerárquica XFL3. Esta especificación puede combinar bloques difusos (para realizar tareas de control y toma de decisiones) y bloques de tipo crisp (para implementar funciones aritméticas y lógicas). Las bases de reglas pueden ser definidas directamente por un operador experto o pueden ser extraídas de datos numéricos utilizando herramientas de identificación y aprendizaje supervisado. La verificación funcional se lleva a cabo analizando la relación entrada–salida del sistema, así como realizando simulaciones de comportamiento en lazo cerrado en combinación con un modelo de la planta descrito en Java. Las herramientas de síntesis hardware que incluye *Xfuzzy* permiten la traducción de una especificación XFL3 a una descripción circuital que puede ser implementada en un dispositivo programable o en un circuito integrado para aplicaciones específicas. Utilizando una herramienta de síntesis incorporada recientemente, xfsg, es posible generar los ficheros requeridos para realizar la estrategia de codiseño que se describe en este trabajo.

2.2. Herramienta de diseño DSP de Xilinx

El entorno SysGen (System Generator) facilita el desarrollo de algoritmos para sistemas de procesado digital de señal (DSP) sobre las FPGAs de Xilinx [13]. Está basado en Simulink, la herramienta interactiva para el modelado, la simulación y el análisis de sistemas dinámicos integrada en Matlab. SysGen incluye una librería de módulos Simulink (denominada “Xilinx Blockset”) que proporciona los elementos básicos de un sistema digital, así como el software necesario para facilitar la síntesis e implementación sobre las FPGAs de Xilinx de los algoritmos descritos y simulados en Matlab.

Los distintos componentes de un sistema se representan en SysGen mediante modelos Simulink que contienen descripciones esquemáticas. La etapa de verificación se lleva a cabo mediante simulación, utilizando las facilidades de generación y visualización de señales ofrecidas por Matlab. Una vez confirmada la funcionalidad del diseño, SysGen facilita su implementación mediante su traslación a diferentes tipos de netlists, la generación del fichero de programación de la FPGA o, incluso, la verificación funcional

del controlador en lazo cerrado a través de la cosimulación hardware-software de la implementación del sistema de control junto a un modelo de la planta. Para llevar a cabo estas tareas, SysGen utiliza las herramientas de síntesis e implementación de FPGAs del entorno ISE de Xilinx.

2.3. Matlab

Un problema esencial en el desarrollo del diseño multilenguaje es la integración de los diferentes sub-sistemas para la implementación del sistema completo. Este problema requiere que se realice una validación efectiva de las funciones del sistema. En este contexto, se decide utilizar Matlab debido a que es una plataforma de desarrollo de alto nivel que permite simulación y puesta a punto de sistemas complejos. Cuenta, además, con buenas herramientas gráficas que posibilitan la visualización de los resultados requeridos.

3. Implementación de controladores difusos sobre FPGAs

La complejidad de los sistemas de control actuales hace necesaria, en ocasiones, la utilización de procesadores de propósito general en combinación con coprocesadores específicos para llevar a cabo las tareas del sistema y asegurar que se cumplen las restricciones de velocidad, tamaño o consumo de energía. Según [5], una solución eficiente para la implementación de un controlador difuso como un sistema híbrido hardware-software consiste en utilizar un procesador de propósito general (que realice las tareas de inicialización, comunicación y control de forma global) en combinación con hardware dedicado en donde se implemente el proceso de inferencia difusa. El controlador descrito en el presente trabajo utiliza el microprocesador MicroBlaze, disponible como módulo de propiedad intelectual (IP) para las FPGAs de Xilinx, junto con módulos de inferencia difusos específicos que permiten la aceleración de las tareas de control.

MicroBlaze es un procesador RISC de 32 bits, con arquitectura Harvard, disponible como módulo-IP para las principales familias de FPGAs de Xilinx. Los buses de direcciones y datos se dividen, a su vez, en buses locales (que facilitan la conexión a los bloques de memoria existentes en la FPGA) y buses de periféricos basados en los estándares OPB y PLB. El entorno de diseño XPS (*Xilinx Platform Studio*) proporciona numerosos periféricos basados en estos buses, así como herramientas que permiten transformar diseños hardware específicos en módulos-IP compatibles con dichos estándares [10]

El diagrama en bloques de los circuitos utilizados para implementar los módulos de inferencia difusa se muestra en la Figura 1. La arquitectura utilizada posibilita una implementación eficiente de un sistema difuso en términos de uso de recursos, consumo de energía y velocidad de ejecución. Las principales características de esta arquitectura son: 1) limitación del grado de solapamiento de las funciones de pertenencia; 2) implementación de una estrategia de procesado que evalúa solo las reglas activas y 3) el uso de métodos de defuzzificación simplificados.

Los circuitos generadores de funciones de pertenencia (MFC) evalúan las entradas y responden con una pareja de valores "etiqueta – nivel de activación" (Li, μi). Un circuito de selección de reglas activas compuesto por un array de multiplexores permite realizar el proceso de inferencia. En cada ciclo de reloj, los grados de pertenencia (μi) de los antecedentes se combinan mediante el operador conectivo de antecedentes para calcular el grado de activación (hi) de la regla. Las etiquetas de los antecedentes (Li) forman la dirección de la memoria de reglas que almacena los parámetros (ci, wi) que definen el consecuente. Para finalizar, el bloque de defuzzificación (DEF) se usa para obtener el valor de salida (y). La temporización del proceso de inferencia se maneja por un bloque de control global [12].

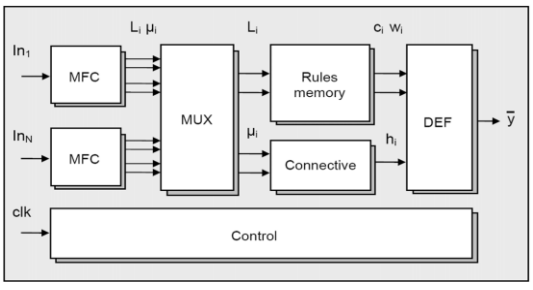


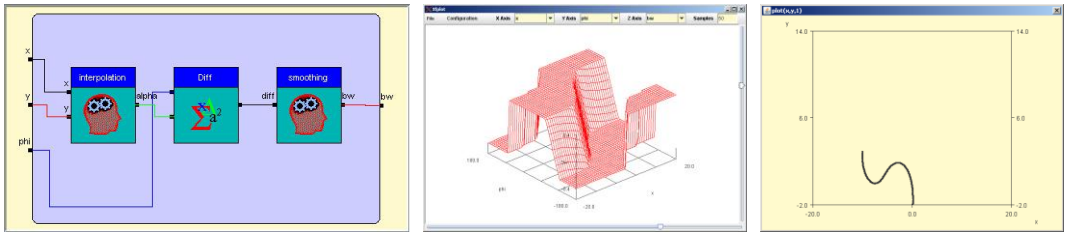
Figura 1. Arquitectura basada en reglas activas [12]

Utilizando SysGen y los bloques contenidos en el grupo “Xilinx Blockset” se ha generado una biblioteca que incluye diferentes alternativas para implementar cada una de los elementos básicos de la arquitectura mostrada en la Figura 1. Los bloques en *XfuzzyLib* están agrupados por su funcionalidad: generadores de funciones de pertenencia, selectores de regla activa, conectivos de antecedentes, memoria de reglas, defuzzificadores y elementos de control. Además de los bloques básicos, la biblioteca incluye elementos que describen a controladores difusos (FLCs) que se diferencian en el número de entradas, el conectivo utilizado para calcular el grado de activación de las reglas y el método de defuzzificación empleado. Actualmente la librería incluye FLCs de 1 y 2 entradas, que utilizan mínimo y producto como conectivo de antecedentes y emplean los métodos de defuzzificación FuzzyMean, WeightFuzzyMean, MaxLabel y Takagi-Sugeno de orden 1.

Los módulos en “Xilinx Blockset” contienen parámetros que permiten que sean definidos aspectos específicos de su funcionalidad, tales como el tamaño del bus o la aritmética a emplear. De igual forma, una vez que el diagrama de bloques de un nuevo elemento ha sido incluido en *XfuzzyLib*, este elemento puede encapsularse como un subsistema y se le puede añadir una máscara que identifique sus parámetros. Cuando este subsistema es utilizado en un nivel de mayor jerarquía los parámetros pueden ser asignados por medio de valores numéricos o por medio de variables. Estas variables pueden ser definidas utilizando la ventana de comandos de Matlab o mediante un fichero “.m”.

3.1. Diseño de alto nivel con *Xfuzzy*

El diseño de un módulo de inferencia con *Xfuzzy* empieza con la descripción del sistema utilizando una especificación XFL3 y las facilidades gráficas que brinda el entorno de desarrollo. La Figura 2-a muestra la descripción gráfica de un controlador difuso jerárquico diseñado para permitir el aparcamiento marcha atrás de un vehículo utilizando una trayectoria óptima [3]. El controlador emplea dos bases de reglas y un bloque crisp. La base de reglas “interpolation” tiene como entrada la posición del vehículo y brinda el valor del ángulo (alpha) asociado con el cambio en el signo de la curvatura. La base de reglas denominada “smoothing” proporciona el valor final de la curvatura que depende de la diferencia entre este ángulo y el definido por la posición del vehículo (phi). Esta base de reglas suaviza las transiciones de las trayectorias, y permite obtener caminos de curvatura continua que pueden ser seguidos por el vehículo sin incumplir sus restricciones dinámicas. Estas bases de reglas han sido definidas y ajustadas utilizando conocimiento experto en combinación con datos de entrenamiento generados a partir de consideraciones geométricas y usados como entrada para la herramienta xfsl incluida en *Xfuzzy*. En la Figura 2-b se puede apreciar la superficie de control del sistema mientras que en la Figura 2-c se ilustra la trayectoria obtenida al realizar una



(a) (b) (c)

Figura 2. a) Representación gráfica de un controlador difuso. b) Superficie de control. c) Simulación en lazo cerrado de un controlador en conjunto con un modelo Java de la planta.

simulación en lazo cerrado del controlador en conjunto con un modelo físico del vehículo codificado en Java.

3.2. Implementación de controladores difusos con SysGen

Construir un sistema utilizando *XfuzzyLib* requiere la identificación e interconexión de los bloques necesarios, así como la definición de los parámetros de acuerdo con el comportamiento que se quiere que estos tengan. Estas tareas se pueden realizar de forma automática por medio de la herramienta de síntesis xfsg incluida en *Xfuzzy*, la cual puede generar un fichero ".mdl" que contiene el modelo Simulink correspondiente a la especificación XFL3 del controlador difuso, así como un fichero ".m" con los parámetros que definen el tamaño y la funcionalidad de los componentes.

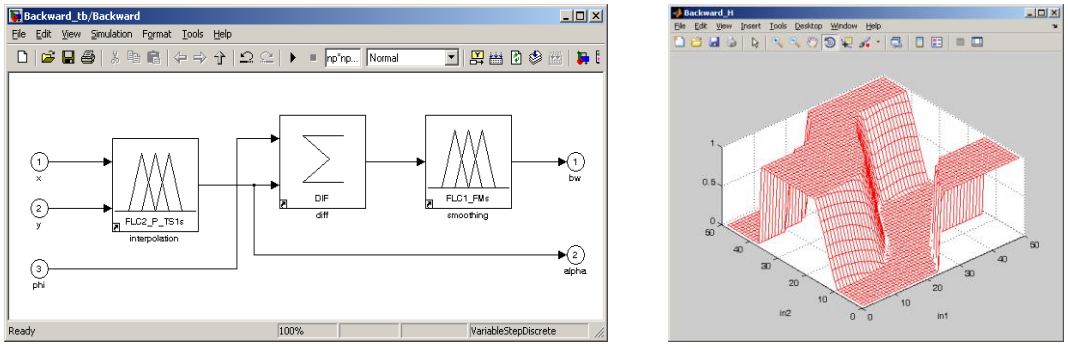
La Figura 3-a muestra el modelo Simulink correspondiente al controlador mostrado en la Figura 2-a. La funcionalidad del diseño puede ser verificada en cualquier momento utilizando las opciones de simulación de Simulink y los bloques correspondientes de generación y muestreo de señales. La superficie de control generada con Matlab se muestra en la Figura 3-b.

Utilizando la herramienta Sysgen es posible realizar el diseño del controlador como un módulo IP conectable a MicroBlaze. Para llevar a cabo este paso es preciso asignar los puertos de entrada/salida y definir su funcionamiento como memoria compartida. A continuación se añade el bloque correspondiente al procesador MicroBlaze. Este bloque contiene la funcionalidad de un proyecto realizado con XPS y generado previamente. Este diseño incluye otros periféricos, como un transmisor/receptor RS-232 y un temporizador. El proyecto debe ser importado hacia el directorio de trabajo de Simulink utilizando las opciones de SysGen que facilitan este proceso.

4. Aplicación al control de un vehículo autónomo

El diseño descrito con anterioridad ha sido utilizado para la realización de un sistema de control difuso para el aparcamiento autónomo de un vehículo eléctrico.

Romeo-4R es un vehículo eléctrico que está equipado con motores de tracción y dirección, así como con diferentes sensores que permiten calcular la posición, orientación y velocidad del vehículo en cada instante [7]. El objetivo del sistema de control a desarrollar es actuar en el sistema de tracción y dirección de forma tal que posibilite el aparcamiento automático a partir de una posición definida. Romeo-4R usa un procesador digital de señales (DSP) TMS-320LF de Texas Instruments que brinda funcionalidades para actuar sobre el control del motor, conversión análogo digital y enlaces de comunicación mediante puerto serie. El DSP adquiere la información de los sensores y estima la posición



(a) (b)

Figura 3. a) Modelo Simulink del controlador mostrado en la Figura 2-a utilizando los FLCs y bloques crisp proporcionados por *XfuzzyLib*. b) Superficie de control obtenida en MatLab

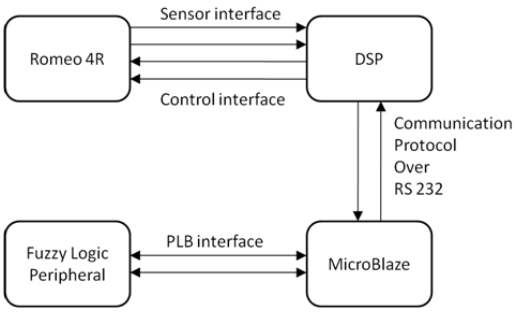


Figura 4. Interfaces de los módulos del sistema

actual del vehículo utilizando un modelo cinemático del mismo [9]. El módulo de control difuso interactúa con el DSP utilizando un protocolo de comunicación por RS-232. El software que se ejecuta en el DSP le proporciona información al módulo de control difuso referente al estado del vehículo y, en respuesta, éste indica los valores deseados de velocidad y ángulo de rotación de las ruedas. (Ver la Figura 4)

El desarrollo conjunto de los componentes hardware y software y sus conexiones se puede ver construido en Simulink como muestra la Figura 5. Este modelo contiene un bloque que es implementado directamente en la FPGA ("Subsystem hwcosim") mientras la simulación se ejecuta. Utilizando esta estrategia se pueden probar distintos modelos de una planta (modelos de vehículos en este caso) y distintos controladores conectados como periféricos a MicroBlaze permitiendo gran flexibilidad de cara al diseñador.

El bloque "Backward" es el módulo de inferencia del controlador difuso correspondiente al modelo de la Figura 3-a, preparado para ser

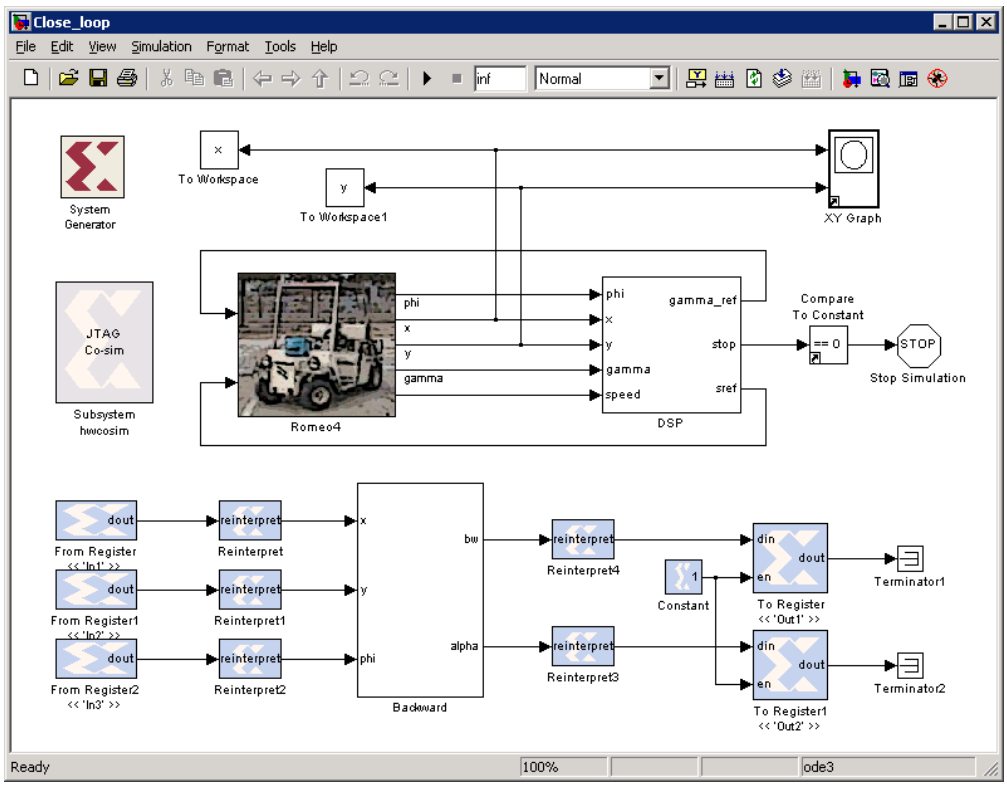


Figura 5. Desarrollo conjunto de componentes hardware y software utilizando la metodología propuesta.

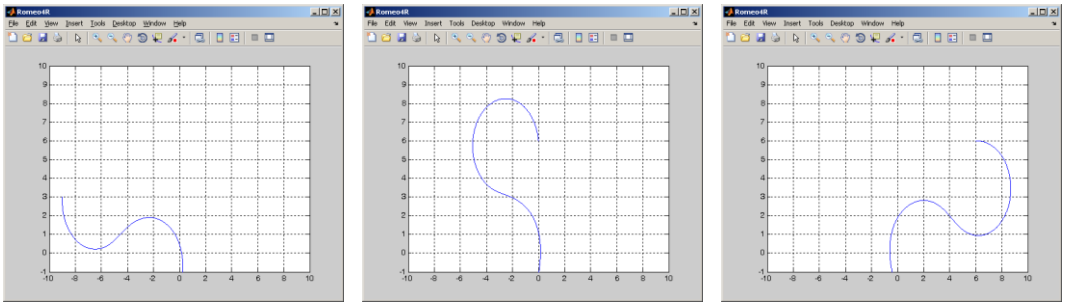


Figura 6. Resultados experimentales que muestran las trayectorias de Romeo-4R para distintas posiciones iniciales

implementado como periférico PLB de MicroBlaze, pero descrito todavía a nivel de modelo para facilitar el ajuste de la base de conocimiento y otros parámetros (el mismo "entorno" serviría para probar otras estrategias de control sin necesidad de volver a implementar la parte del sistema que ya está en la FPGA.

El bloque "Subsystem hwcosim" representa al procesador implementado en la placa. Está preparado para incorporar un periférico de lógica difusa en cualquiera de sus interfaces Facilita también una interfaz para el desarrollo de las aplicaciones (lazo de control y comunicación con el DSP a través de RS-232 y un protocolo ad-hoc).

El protocolo de comunicación y los controladores software para realizar la comunicación con el periférico FLC son ejecutados por el bloque "Subsystem hwcosim" previa compilación. SysGen está preparada para cosimular las funciones que están implementadas en la FPGA junto con el resto de la funcionalidad que está descrita en el modelo Simulink de la Figura 5. Para la representación del sistema completo fue necesario utilizar otros modelos. El bloque "Romeo4R" es una representación de la funcionalidad del vehículo basada en un modelo matemático del mismo. El bloque "DSP" emula al procesador correspondiente y posibilita la interacción entre el controlador difuso y el vehículo. Este modelo incorpora la capacidad de comunicación por RS-232 para transmitir información hasta el procesador que está implementado en el FPGA. También se incluye un bloque personalizado de tipo "S-Function Block" que permite la descripción del DSP utilizando una implementación de tipo "S-Function" en Matlab. El proceso de creación de esta "S-Function" se realiza utilizando ficheros de plantilla que proporciona Matlab en conjunto con la documentación correspondiente. Para inicializar parámetros y visualizar el contenido de algunos puerto han sido añadidos al modelo otros bloques de Simulink. En la Figura 6 se incluyen los resultados experimentales que ilustran las trayectorias del vehículo para distintas posiciones iniciales.

Cuando la funcionalidad del sistema ha sido verificada, todo el sistema puede ser implementado en la FPGA utilizando el bloque "System Generator" que se incorpora en el modelo de la Figura 5. Se usaron 12 bits como resolución de la entrada y salida del módulo de inferencia y como resultado se obtuvo un sistema que consume el 14% de los recursos disponibles en el FPGA (XC3S700).

5. Conclusiones

En este trabajo se ha presentado una estrategia para la realización de controladores difusos empotrados en FPGAs. Dicha metodología está basada en un diseño multilenguaje orientado hacia la cosimulación. Esta realización ha permitido el desarrollo de un sistema híbrido hardware-software que combina el uso de un procesador de propósito general y módulos de inferencia difusa.

El uso de un flujo de diseño utilizando distintas herramientas CAD e integrándolo en Matlab ha permitido verificar el comportamiento del diseño en un lazo cerrado con el modelo de la planta en cuestión, a la vez que se ha podido sintonizar de forma concurrente sus parámetros y verificar los componentes descritos tanto para hardware como para software.

Con la intención de demostrar las ventajas que este tipo de realización presenta, se ha aplicado esta estrategia de diseño a la realización de un controlador difuso para el aparcamiento de un vehículo autónomo.

Referencias

[1] C. Ajluni, "System-Level Languages Fight To Take Over As The Next Design Solution", Electronic Design Automation, vol. 48, n. 2, Jan. 2000.

[2] S. Akira Ito, et al., "System Design Based on Single Language and Single-Chip Java ASIP Microcontroller", Design, Automation, and Test in Europe, France, pp. 703 - 709. 2000.

[3] I. Baturone el al., "Automatic design of fuzzy controllers for car-like autonomous robots", IEEE Trans. on Fuzzy Systems, vol. 12, n. 4, pp. 447-465. Aug. 2004.

[4] A. Bunker, et al., "An Overview of Formal Hardware Specification Languages", 4th ACM/CRA Grace Hopper Celebration of Women in Computing, San Diego, CA, 2002.

[5] A. Cabrera et al., "Hardware/software co-design of configurable fuzzy control systems", Applied Soft Computing, vol. 4, n. 3, pp. 271-285, Aug. 2004.

[6] P. Coste, et al., "Multilanguage Design of Heterogeneous Systems", Int. Workshop on Hardware/Software Codesign, 1999.

[7] F. Cuesta et al., "Parking maneuvers of industrial-like electrical vehicles with and without trailer", IEEE Trans. on Industrial Electronics, vol. 51, n. 2, pp. 257-269, Apr. 2004.

[8] M. Dorfman and R. H. Thayer, "Standards, Guidelines, and Examples on Systems and Software Requirements Engineering", IEEE Computer Society Press.

[9] J. Ferruz et al., "An embedded DSP-based controller for the Romeo-4R vehicle", IFAC Int. Symp. On Intelligent Components and Instruments for Control Applications, Jul. 2003.

[10] MicroBlaze Reference Guides, Xilinx, Inc.

[11] I. Page, "Constructing Hardware-Software Systems from a single Description", Journal of VLSI Signal Processing, vol. 12, pp. 87-107, 1996.

[12] S. Sánchez-Solano et al., "FPGA Implementation of Embedded Fuzzy Controllers for Robotic Applications". IEEE Trans. on Industrial Electronics, vol. 54, n. 4, pp. 1937-1945, Aug. 2007.

[13] System Generator for DSP User Guide, v10.1, Xilinx Inc., 2008. Available: http://www.xilinx.com

[14] Xfuzzy: Fuzzy Logic Design Tools, IMSE-CNM, CSIC. Available: http://www.imse-cnm.csic.es/Xfuzzy